IST 634 ENTERPRISE DATABASES

FALL - 2024

SECTION-1

GROUP – 4

Team Members:

Aditya Sai Aravind Bodaka - 2886068

Sathvika Adam - 2886305

Jaya Krishna Sadhu Muni – 2886381

Anil Kumar Dharavath – 2885797

Build Smart Pro: Comprehensive Construction Management Application

Company Overview

BuildSmart is a comprehensive construction management tool created to streamline and streamline a construction company's activities. It combines essential elements of financial tracking, staff distribution, supplier coordination, customer relations, and project management into a unified platform. For large-scale projects, BuildSmart enables the business to effectively manage budgets, schedules, and resources while guaranteeing real-time tracking of invoices, past-due payments, and material usage. It improves decision-making and resource use by offering comprehensive insights into staff assignments, supplier performance, and project status. By reducing errors, streamlining processes, and guaranteeing financial transparency, this integrated strategy enables the business to complete high-quality construction projects on schedule and within budget.

As the foundation of the business's operations, the BuildSmart program is based on a strong enterprise-grade relational database. It makes data management easier and enhances decision-making in a variety of departments. BuildSmart enables the business to accomplish its operational objectives while adjusting to the intricacies of the construction sector by combining all essential components of construction management into a single, unified platform.

A powerful relational database is used by the BuildSmart software to consolidate and expedite construction management processes.

According to the database, this is how it works, step-by-step:

Centralized data storage:

- Structured database tables contain all of the important elements, including clients, projects, employees, suppliers, invoices, contracts, and project roles.
- Assuring relationships between entities, these tables are linked using primary and foreign keys (e.g., a project is linked to its customer, employees, and bills).

Project Management:

- Details like the project name, budget, schedule, and related client are kept in the Projects table.
- Through the app, customers can keep tabs on project progress, expenditures, and resource allocation.
- Automated queries can be used to identify any overspending or past-due assignments.

Client and Contract Management:

- Client data is kept in the Clients table, which makes sure that each project is connected to the appropriate client.
- Project-specific agreements, complete with terms and signed dates, are stored in the Contracts database for compliance and convenience.

Employee Assignment:

- Staff information, positions, and certificates are kept in the Employees database.
- The Project Roles database keeps track of the beginning and ending dates of assignments and associates' workers with particular projects.
- This guarantees effective administration of the workforce and responsibility for every position.

Supplier and Inventory Coordination:

- Vendor information, ratings, and materials supplied are managed in the Suppliers table.
- By keeping track of supplies, amounts, and supplier relationships, the inventory table helps avoid shortages and building delays.

Financial Management:

- All financial transactions, including amounts, due dates, and payment statuses (such as Paid, Pending, and Overdue), are tracked in the Invoices table.
- To ensure financial transparency, queries can determine the overall cost of a project, highlight past-due payments, or spot income trends over time.

Reports and Insights:

- Clients with past-due payments are among the insights produced by the app.
- projects that go beyond the budget.
- top-performing personnel or suppliers.
- Decision-making is guided by these insights, which also increase operational effectiveness.

Automation and Alerts:

- Real-time updates for past-due invoices, low inventory levels, or unassigned personnel are guaranteed via automated queries and notifications.
- This proactive strategy guarantees seamless operations and reduces delays.

How BuildSmart Uses the Database:

BuildSmart offers a smooth experience by using relationships to connect all these tables, where:

- Tasks can be assigned, progress can be tracked, and expenses can be tracked by managers.
- Financial reports and payments can be managed by accountants.
- Materials and suppliers can be managed by procurement teams.
- The database serves as the app's fundamental building block, guaranteeing accurate, consistent, and real-time data for all construction management procedures.

Challenges:

- Managing multiple ongoing projects.
- Tracking inventory and procurement.
- Ensuring timely completion and cost control.
- Maintaining client and supplier relationships.

Benefits of BuildSmart Constructions Inc.:

- **Operational Efficiency**: Automated procedures and streamlined workflows minimize manual labor and save time.
- **Making Informed Decisions:** Better planning and resource allocation are supported by real-time access to reliable data.
- **Customer satisfaction:** Trust and loyalty are increased through prompt communication and project updates.
- **Growth and Scalability:** As BuildSmart grows, the database may grow to accommodate new clients and projects.

Database Design

Entities in the Database

1. Clients: Represents the customers of the company.
2. Projects: Represents the work or assignments undertaken for clients.
3. Invoices: Represents billing other payment related information for projects.
4. Suppliers: Represents vendors providing the materials or services.
5. Inventory: Represents items in stock that are managed by the company.
6. Contracts: Represents the agreements made between clients and the company.

7. Employees: Represents the company's staff.
8. ProjectRoles: Represents the assignment of employees to specific projects.

Entity Relationships and Cardinalities

1. Clients and Projects

- Relationship: A client can have multiple projects, but a project belongs to only one client.
- Cardinality: One-to-Many
- Reason: Each client can request multiple projects, but a project is tied to a single client for accountability and billing.

2. Projects and Invoices

- Relationship: A project can have multiple invoices, but an invoice is associated with a single project.
- Cardinality: One-to-Many
- Reason: Billing for a project might be divided into multiple phases or milestones, requiring multiple invoices.

3. Projects and Contracts

- Relationship: A project has one associated contract, and a contract is tied to a single project.
- Cardinality: 1:1/One-to-One
- Reason: Each project is bound by a unique and strong agreement to define its terms and conditions.

4. Projects and Employees

- Relationship: A project can involve multiple employees, and an employee can work on multiple projects.
- Cardinality: Many-to-Many
- Reason: Employees can be assigned to various projects in different roles, and projects require contributions from many employees.

5. Suppliers and Inventory

- Relationship: A supplier can supply multiple inventory items, but an inventory item is supplied by one supplier.

- Cardinality: One-to-Many
- Reason: Suppliers provide a range of items, but each item is usually sourced from a primary supplier to simplify supply chain management.

6. Employees and ProjectRoles

- Relationship: An employee can have multiple roles across different projects, and each role is associated with a single employee.
- Cardinality: One-to-Many
- Reason: Employees can take on various roles (e.g., manager, developer) across different projects.

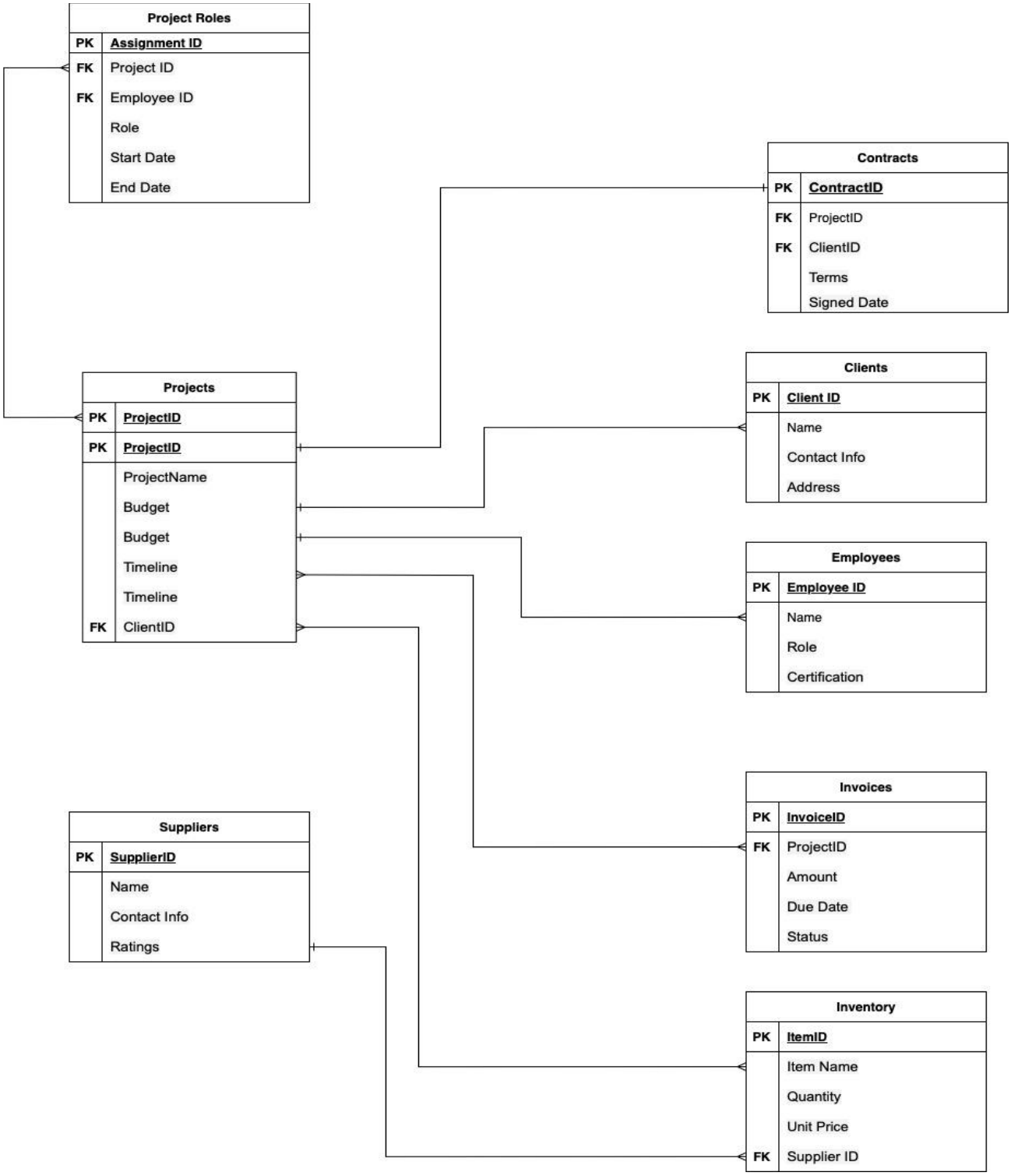
7. Contracts and Clients

- Relationship: A client can have multiple contracts, but a contract is tied to a single client.
- Cardinality: One-to-Many
- Reason: Clients may have ongoing or separate agreements for different projects.

8. Invoices and Inventory

- Relationship: An invoice can reference multiple inventory items used for a project.
- Cardinality: Many-to-Many
- Reason: Projects often require materials from the inventory, and an invoice may consolidate charges for multiple items.

ER Diagram


Project Roles
PK Assignment ID
FK Project ID
FK Employee ID
Role
Start Date
End Date
Contracts
PK ContractID
FK ProjectID
FK ClientID
Terms
Signed Date
Projects
PK ProjectID
PK ProjectID
ProjectName
Budget
Budget
Timeline
Timeline
FK ClientID
Clients
PK Client ID
Name
Contact Info
Address
Employees
PK Employee ID
Name
Role
Certification
Invoices
PK InvoiceID
FK ProjectID
Amount
Due Date
Status
Suppliers
PK SupplierID
Name
Contact Info
Ratings
Inventory
PK ItemID
Item Name
Quantity
Unit Price
FK Supplier ID

Normalization:

1NF (First Normal Form)

It ensures that all columns in a table contain atomic (indivisible) values and that there are no repeating groups or arrays.

Analysis of each Tables:

1. Clients Table:
    - Columns: ClientID, Name, ContactInfo, Address
    - Satisfies 1NF: All fields contain atomic values (e.g., Name is a single value, not a list of names).
2. Projects Table:
    - Columns: ProjectID, ProjectName, Budget, Timeline, ClientID
    - Satisfies 1NF: All fields are atomic. No repeating groups or arrays.
3. Invoices Table:
    - Columns: InvoiceID, ProjectID, Amount, DueDate, Status
    - Satisfies 1NF: Each invoice is unique, and all values are atomic.

2NF (Second Normal Form)

It ensures that:

1. It satisfies 1NF.
2. All non-key attributes are fully functionally dependent on the entire primary key (no partial dependency).

Analysis of the Tables:

1. Clients Table:
    - Primary Key: ClientID
    - Non-key attributes: Name, ContactInfo, Address
    - Satisfies 2NF: All non-key attributes depend on ClientID.

2. Projects Table:
    - Primary Key: ProjectID
    - Foreign Key: ClientID
    - Non-key attributes: ProjectName, Budget, Timeline
    - Satisfies 2NF: All non-key attributes depend on ProjectID, not just ClientID.
3. Invoices Table:
    - Primary Key: InvoiceID
    - Foreign Key: ProjectID
    - Non-key attributes: Amount, DueDate, Status
    - Satisfies 2NF: All non-key attributes depend on InvoiceID.

3NF (Third Normal Form)

It ensures that:

1. The database satisfies 2NF.
2. All non-key attributes are non-transitively dependent on the primary key (i.e., no transitive dependencies).

Analysis of the Tables:

1. Clients Table:
    - Primary Key: ClientID
    - Non-key attributes: Name, ContactInfo, Address
    - Satisfies 3NF: No transitive dependencies. Attributes depend only on ClientID.
2. Projects Table:
    - Primary Key: ProjectID
    - Foreign Key: ClientID
    - Non-key attributes: ProjectName, Budget, Timeline
    - Satisfies 3NF: No transitive dependencies. Attributes depend directly on ProjectID.

3. Invoices Table:
    - Primary Key: InvoiceID
    - Foreign Key: ProjectID
    - Non-key attributes: Amount, DueDate, Status
    - Satisfies 3NF: No transitive dependencies. Attributes depend only on InvoiceID.
4. Suppliers and Inventory Tables:
    - Primary Key: SupplierID (Suppliers), ItemID (Inventory)
    - Foreign Key: SupplierID in Inventory
    - Satisfies 3NF: No transitive dependencies. Each attribute depends directly on its primary key.

Database Creation:

SQL Scripts:

Database Creation:

```
create database Final_Project;

use Final_Project;
```

Creating Table:

```
-- Clients Table
CREATE TABLE Clients (
    ClientID VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    ContactInfo VARCHAR(15) NOT NULL,
    Address VARCHAR(255) NOT NULL
);


-- Suppliers Table
CREATE TABLE Suppliers (
```

```
    SupplierID VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    ContactInfo VARCHAR(15) NOT NULL,
    Ratings DECIMAL(2, 1) CHECK (Ratings BETWEEN 1.0 AND 5.0)
);

-- Employees Table
CREATE TABLE Employees (
    EmployeeID VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Role VARCHAR(50) NOT NULL,
    Certification VARCHAR(50)
);

-- Projects Table
CREATE TABLE Projects (
    ProjectID VARCHAR(10) PRIMARY KEY,
    ProjectName VARCHAR(100) NOT NULL,
    Budget DECIMAL(15, 2) NOT NULL,
    Timeline VARCHAR(20) NOT NULL,
    ClientID VARCHAR(10),
    FOREIGN KEY (ClientID) REFERENCES Clients(ClientID)
);

-- Inventory Table
CREATE TABLE Inventory (
    ItemID VARCHAR(10) PRIMARY KEY,
```

```
    ItemName VARCHAR(100) NOT NULL,
    Quantity INT NOT NULL CHECK (Quantity >= 0),
    UnitPrice DECIMAL(10, 2) NOT NULL,
    SupplierID VARCHAR(10),
    FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)
);

-- Invoices Table
CREATE TABLE Invoices (
    InvoiceID VARCHAR(10) PRIMARY KEY,
    ProjectID VARCHAR(10),
    Amount DECIMAL(10, 2) NOT NULL,
    DueDate DATE NOT NULL,
    Status VARCHAR(20) CHECK (Status IN ('Paid', 'Pending', 'Overdue')),
    FOREIGN KEY (ProjectID) REFERENCES Projects(ProjectID)
);

-- Contracts Table
CREATE TABLE Contracts (
    ContractID VARCHAR(10) PRIMARY KEY,
    ProjectID VARCHAR(10),
    ClientID VARCHAR(10),
    Terms VARCHAR(255) NOT NULL,
    SignedDate DATE NOT NULL,
    FOREIGN KEY (ProjectID) REFERENCES Projects(ProjectID),
    FOREIGN KEY (ClientID) REFERENCES Clients(ClientID)
);
```

```
-- Project Roles Table
CREATE TABLE ProjectRoles (
    AssignmentID VARCHAR(10) PRIMARY KEY,
    ProjectID VARCHAR(10),
    EmployeeID VARCHAR(10),
    Role VARCHAR(50) NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE,
    FOREIGN KEY (ProjectID) REFERENCES Projects(ProjectID),
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

Data Population:

```
-- Clients Table

INSERT INTO Clients (ClientID, Name, ContactInfo, Address) VALUES
('C101', 'Urban Builders Inc.', '212-555-0112', '123 Broadway, New York, NY 10001'),
('C102', 'EcoHomes LLC', '312-555-0199', '456 Lakeshore Ave, Chicago, IL 60601'),
('C103', 'Pinnacle Properties', '415-555-0123', '789 Market St, San Francisco, CA 94103'),
('C104', 'Metro Real Estate Group', '702-555-0145', '321 Fremont Ave, Las Vegas, NV 89101'),
('C105', 'Bay Area Ventures', '510-555-0167', '555 First St, Oakland, CA 94607'),
('C106', 'Skyline Development', '213-555-0178', '555 Sunset Blvd, Los Angeles, CA 90028'),
('C107', 'Blue Horizon Estates', '503-555-0193', '678 Elm St, Portland, OR 97201'),
('C108', 'Greenway Partners', '617-555-0143', '789 Oak St, Boston, MA 02116'),
('C109', 'Riverfront Holdings', '614-555-0168', '901 Water St, Columbus, OH 43215'),
('C110', 'Sunset Realty', '480-555-0184', '234 Main St, Phoenix, AZ 85001'),
```

('C111', 'Oceanview Group', '305-555-0179', '101 Ocean Ave, Miami, FL 33101'),

('C112', 'Crestfield Construction', '214-555-0188', '200 Maple St, Dallas, TX 75201'),

('C113', 'Golden Gate Development', '415-555-0174', '300 Bay St, San Francisco, CA 94133'),

('C114', 'Aspen Builders', '303-555-0192', '400 Aspen Dr, Denver, CO 80203'),

('C115', 'Highland Constructors', '312-555-0176', '500 Lake St, Chicago, IL 60611'),

('C116', 'Valley View Estates', '602-555-0187', '600 Mountain Rd, Phoenix, AZ 85018'),

('C117', 'Parkside Residences', '404-555-0183', '700 Park Ave, Atlanta, GA 30303'),

('C118', 'Sunrise Communities', '210-555-0190', '800 Spring St, San Antonio, TX 78201'),

('C119', 'Lakeside Homes', '919-555-0191', '900 Hill St, Raleigh, NC 27601'),

('C120', 'Midtown Developers', '213-555-0185', '100 Midtown Ave, Los Angeles, CA 90013'),

('C121', 'NorthStar Builders', '503-555-0177', '110 North Ave, Portland, OR 97209'),

('C122', 'Westfield Construction', '312-555-0182', '120 West St, Chicago, IL 60602'),

('C123', 'Maplewood Homes', '720-555-0171', '130 Maple Dr, Boulder, CO 80301'),

('C124', 'Brighton Estates', '214-555-0195', '140 Pine St, Austin, TX 78701'),

('C125', 'Harmony Developments', '510-555-5678', '123 Oak St, Berkeley, CA 94720');

-- Suppliers Table

INSERT INTO Suppliers (SupplierID, Name, ContactInfo, Ratings) VALUES

('S101', 'Concrete Solutions Inc.', '212-555-0199', 4.7),

('S102', 'Green Lumber Supply', '617-555-0101', 4.5),

('S103', 'Industrial Steel Co.', '312-555-0188', 4.8),

('S104', 'Builders Brickworks', '713-555-0143', 4.3),

('S105', 'Glass and Glazing Corp.', '415-555-0155', 4.6),

('S106', 'Advanced Building Supply', '323-555-5678', 4.4),

('S107', 'Eco Concrete Supplies', '202-555-0189', 4.2),

('S108', 'Highland Timber', '509-555-0123', 4.6),

('S109', 'Allied Steelworks', '404-555-0145', 4.7),

('S110', 'BrickMaster LLC', '305-555-0181', 4.5),

('S111', 'Custom Glassworks', '608-555-0192', 4.4),

('S112', 'Pacific Lumber Co.', '310-555-0163', 4.3),

('S113', 'Midwest Concrete', '317-555-0174', 4.6),

('S114', 'Metro Metal Supply', '916-555-0128', 4.8),

('S115', 'Lumbertown Supplies', '503-555-0193', 4.3),

('S116', 'Precision Stoneworks', '402-555-0139', 4.5),

('S117', 'Durable Concrete', '415-555-0115', 4.2),

('S118', 'National Rebar', '213-555-0144', 4.6),

('S119', 'Fine Finish Glass', '650-555-0187', 4.7),

('S120', 'Evergreen Lumber', '702-555-0158', 4.5),

('S121', 'Cornerstone Materials', '919-555-0166', 4.4),

('S122', 'Central Concrete Co.', '615-555-0191', 4.3),

('S123', 'Top Quality Brick', '808-555-0179', 4.8),

('S124', 'Glassworks USA', '212-555-0125', 4.6),

('S125', 'Timberland Suppliers', '213-555-0194', 4.4);

-- Employees Table

INSERT INTO Employees (EmployeeID, Name, Role, Certification) VALUES

('E101', 'Michael Johnson', 'Senior Structural Engineer', 'Licensed PE'),

('E102', 'Samantha Lee', 'Project Manager', 'Certified PM'),

('E103', 'Alex Rodriguez', 'Field Supervisor', 'OSHA Level 2'),

('E104', 'Linda Chang', 'Principal Architect', 'Registered Architect'),

('E105', 'David Kim', 'Electrician', 'Journeyman Electrician'),

('E106', 'Jennifer Green', 'Site Engineer', 'Certified Engineer'),

('E107', 'Robert Clark', 'Carpenter', 'Certified Carpenter'),

('E108', 'Nancy Thomas', 'Plumber', 'Master Plumber'),

('E109', 'Kevin Brown', 'Mechanical Engineer', 'PE Mechanical'),

('E110', 'Angela White', 'Civil Engineer', 'Licensed PE'),

('E111', 'Thomas Harris', 'Construction Manager', 'PMP Certified'),

('E112', 'Maria Lewis', 'Surveyor', 'Certified Surveyor'),

('E113', 'Joseph Wilson', 'Architect', 'LEED Accredited'),

('E114', 'Karen Walker', 'Structural Engineer', 'Licensed PE'),

('E115', 'Richard Allen', 'Safety Officer', 'OSHA Certified'),

('E116', 'Laura Scott', 'Interior Designer', 'NCIDQ Certified'),

('E117', 'Daniel Young', 'Estimator', 'Certified Estimator'),

('E118', 'Jessica King', 'Project Coordinator', 'Certified Coordinator'),

('E119', 'James Wright', 'Electrical Engineer', 'Licensed PE'),

('E120', 'Patricia Perez', 'HVAC Technician', 'Certified Technician'),

('E121', 'Brian Hall', 'Drafter', 'Certified Drafter'),

('E122', 'Susan Green', 'Foreman', 'Experienced Foreman'),

('E123', 'Steven Adams', 'Laborer', 'Experienced Laborer'),

('E124', 'Emma Miller', 'Welder', 'Certified Welder'),

('E125', 'Olivia Davis', 'Environmental Engineer', 'Green Certification');

-- Projects Table

INSERT INTO Projects (ProjectID, ProjectName, Budget, Timeline, ClientID) VALUES

('P101', 'City Center Mall Renovation', 780000.00, '12 months', 'C101'),

('P102', 'Greenway Apartments Development', 1250000.00, '24 months', 'C102'),

('P103', 'Riverside Park Office Tower', 2100000.00, '30 months', 'C103'),

('P104', 'Lakeside Community Housing', 460000.00, '15 months', 'C104'),

('P105', 'Sunset Villas Expansion', 965000.00, '20 months', 'C105'),

('P106', 'Oceanview Condos', 1450000.00, '18 months', 'C106'),

('P107', 'Downtown Office Plaza', 2750000.00, '24 months', 'C107'),

('P108', 'Highland Park Retail Center', 1950000.00, '22 months', 'C108'),

('P109', 'Midtown Tech Campus', 3200000.00, '30 months', 'C109'),

('P110', 'Valley View Estates', 1250000.00, '20 months', 'C110'),

('P111', 'Riverbend Apartments', 850000.00, '12 months', 'C111'),

('P112', 'Grandview Shopping Center', 1400000.00, '18 months', 'C112'),

('P113', 'Bay Ridge Business Park', 2700000.00, '25 months', 'C113'),

('P114', 'Southside Townhomes', 900000.00, '14 months', 'C114'),

('P115', 'Parkside Luxury Towers', 3200000.00, '28 months', 'C115'),

('P116', 'Horizon Medical Center', 2800000.00, '24 months', 'C116'),

('P117', 'Summit Plaza Expansion', 1100000.00, '18 months', 'C117'),

('P118', 'North End Office Park', 2450000.00, '26 months', 'C118'),

('P119', 'Westside Family Housing', 750000.00, '14 months', 'C119'),

('P120', 'Eastview Residential Complex', 1900000.00, '20 months', 'C120'),

('P121', 'Central Park Redevelopment', 3000000.00, '36 months', 'C121'),

('P122', 'Maplewood Villas', 1400000.00, '18 months', 'C122'),

('P123', 'Heritage Senior Living', 1300000.00, '16 months', 'C123'),

('P124', 'Brighton Lofts', 1500000.00, '22 months', 'C124'),

('P125', 'Harmony Gardens Condos', 1750000.00, '24 months', 'C125');

-- Inventory Table

INSERT INTO Inventory (ItemID, ItemName, Quantity, UnitPrice, SupplierID) VALUES

('I101', 'Ready-Mix Concrete', 1200, 85.00, 'S101'),

('I102', 'Structural Lumber', 700, 68.00, 'S102'),

('I103', 'Reinforcing Steel', 500, 130.00, 'S103'),

('I104', 'Clay Bricks', 10000, 0.80, 'S104'),

('I105', 'Tempered Glass', 300, 160.00, 'S105'),

('I106', 'High-Strength Cement', 500, 90.00, 'S106'),

('I107', 'Pine Lumber', 1000, 75.00, 'S107'),

('I108', 'Steel Beams', 250, 200.00, 'S108'),

('I109', 'Roof Tiles', 1200, 2.50, 'S109'),

('I110', 'Insulation Panels', 400, 45.00, 'S110'),

('I111', 'Copper Pipes', 800, 30.00, 'S111'),

('I112', 'Glass Windows', 200, 150.00, 'S112'),

('I113', 'Drywall Sheets', 500, 10.00, 'S113'),

('I114', 'Ceramic Tiles', 1000, 5.50, 'S114'),

('I115', 'Concrete Blocks', 1500, 1.20, 'S115'),

('I116', 'Rebar Steel', 600, 135.00, 'S116'),

('I117', 'Flooring Panels', 300, 25.00, 'S117'),

('I118', 'PVC Pipes', 1200, 2.20, 'S118'),

('I119', 'Hardwood Flooring', 400, 55.00, 'S119'),

('I120', 'Asphalt Shingles', 900, 3.00, 'S120'),

('I121', 'Plumbing Fittings', 500, 15.00, 'S121'),

('I122', 'Granite Countertops', 100, 250.00, 'S122'),

('I123', 'Sand and Gravel', 3000, 0.30, 'S123'),

('I124', 'Metal Roofing', 500, 80.00, 'S124'),

('I125', 'Aluminum Siding', 350, 70.00, 'S125');

-- Invoices Table

INSERT INTO Invoices (InvoiceID, ProjectID, Amount, DueDate, Status) VALUES

('INV101', 'P101', 10000.00, '2024-02-01', 'Paid'),

('INV102', 'P102', 50000.00, '2024-03-15', 'Pending'),

('INV103', 'P103', 85000.00, '2024-04-01', 'Overdue'),

('INV104', 'P104', 13500.00, '2024-05-01', 'Paid'),

('INV105', 'P105', 16000.00, '2024-05-15', 'Pending'),

('INV106', 'P106', 22000.00, '2024-06-01', 'Paid'),

('INV107', 'P107', 50000.00, '2024-06-15', 'Pending'),

('INV108', 'P108', 30000.00, '2024-07-01', 'Paid'),

('INV109', 'P109', 45000.00, '2024-07-15', 'Overdue'),

('INV110', 'P110', 28000.00, '2024-08-01', 'Paid'),

('INV111', 'P111', 37000.00, '2024-08-15', 'Pending'),

('INV112', 'P112', 29000.00, '2024-09-01', 'Paid'),

('INV113', 'P113', 41000.00, '2024-09-15', 'Pending'),

('INV114', 'P114', 45000.00, '2024-10-01', 'Paid'),

('INV115', 'P115', 32000.00, '2024-10-15', 'Overdue'),

('INV116', 'P116', 51000.00, '2024-11-01', 'Paid'),

('INV117', 'P117', 40000.00, '2024-11-15', 'Pending'),

('INV118', 'P118', 27000.00, '2024-12-01', 'Paid'),

('INV119', 'P119', 15000.00, '2024-12-15', 'Paid'),

('INV120', 'P120', 43000.00, '2025-01-01', 'Pending'),

('INV121', 'P121', 49000.00, '2025-01-15', 'Paid'),

('INV122', 'P122', 22000.00, '2025-02-01', 'Pending'),

('INV123', 'P123', 14000.00, '2025-02-15', 'Overdue'),

('INV124', 'P124', 33000.00, '2025-03-01', 'Paid'),

('INV125', 'P125', 29000.00, '2025-03-15', 'Pending'),

('INV126', 'P101', 27000.00, '2025-04-01', 'Paid'),

('INV127', 'P102', 24000.00, '2025-04-15', 'Overdue'),

('INV128', 'P103', 30000.00, '2025-05-01', 'Paid'),

('INV129', 'P104', 17000.00, '2025-05-15', 'Pending'),

('INV130', 'P105', 35000.00, '2025-06-01', 'Paid');

-- Contracts Table

INSERT INTO Contracts (ContractID, ProjectID, ClientID, Terms, SignedDate) VALUES

('CTR101', 'P101', 'C101', 'Lump Sum Contract - Full Payment on Completion', '2024-01-10'),

('CTR102', 'P102', 'C102', 'Cost Plus Contract - Monthly Billing for Expenses', '2024-02-12'),

('CTR103', 'P103', 'C103', 'Fixed Price Contract - 30% Advance', '2024-03-05'),

('CTR104', 'P104', 'C104', 'Guaranteed Maximum Price Contract', '2024-04-01'),

('CTR105', 'P105', 'C105', 'Unit Price Contract - Per Square Foot', '2024-05-01'),

('CTR106', 'P106', 'C106', 'Lump Sum Contract - Full Payment on Completion', '2024-06-01'),

('CTR107', 'P107', 'C107', 'Cost Plus Contract - Monthly Billing for Expenses', '2024-07-10'),

('CTR108', 'P108', 'C108', 'Fixed Price Contract - 25% Advance', '2024-08-15'),

('CTR109', 'P109', 'C109', 'Guaranteed Maximum Price Contract', '2024-09-01'),

('CTR110', 'P110', 'C110', 'Unit Price Contract - Per Square Foot', '2024-10-01'),

('CTR111', 'P111', 'C111', 'Lump Sum Contract - Full Payment on Completion', '2024-11-01'),

('CTR112', 'P112', 'C112', 'Cost Plus Contract - Monthly Billing for Expenses', '2024-12-01'),

('CTR113', 'P113', 'C113', 'Fixed Price Contract - 20% Advance', '2025-01-01'),

('CTR114', 'P114', 'C114', 'Guaranteed Maximum Price Contract', '2025-02-01'),

('CTR115', 'P115', 'C115', 'Unit Price Contract - Per Square Foot', '2025-03-01'),

('CTR116', 'P116', 'C116', 'Lump Sum Contract - Full Payment on Completion', '2025-04-01'),

('CTR117', 'P117', 'C117', 'Cost Plus Contract - Monthly Billing for Expenses', '2025-05-01'),

('CTR118', 'P118', 'C118', 'Fixed Price Contract - 30% Advance', '2025-06-01'),

('CTR119', 'P119', 'C119', 'Guaranteed Maximum Price Contract', '2025-07-01'),

('CTR120', 'P120', 'C120', 'Unit Price Contract - Per Square Foot', '2025-08-01'),

('CTR121', 'P121', 'C121', 'Lump Sum Contract - Full Payment on Completion', '2025-09-01'),

('CTR122', 'P122', 'C122', 'Cost Plus Contract - Monthly Billing for Expenses', '2025-10-01'),

('CTR123', 'P123', 'C123', 'Fixed Price Contract - 25% Advance', '2025-11-01'),

('CTR124', 'P124', 'C124', 'Guaranteed Maximum Price Contract', '2025-12-01'),

('CTR125', 'P125', 'C125', 'Unit Price Contract - Per Square Foot', '2026-01-01'),

('CTR126', 'P101', 'C101', 'Fixed Price Contract - 15% Advance', '2026-02-01'),

('CTR127', 'P102', 'C102', 'Guaranteed Maximum Price Contract', '2026-03-01'),

('CTR128', 'P103', 'C103', 'Unit Price Contract - Per Square Meter', '2026-04-01'),

('CTR129', 'P104', 'C104', 'Lump Sum Contract - Full Payment on Completion', '2026-05-01'),

('CTR130', 'P105', 'C105', 'Cost Plus Contract - Billing with Receipts', '2026-06-01');

-- Project Roles Table

INSERT INTO ProjectRoles (AssignmentID, ProjectID, EmployeeID, Role, StartDate, EndDate) VALUES

('R101', 'P101', 'E101', 'Structural Engineer', '2024-01-15', '2024-07-15'),

('R102', 'P102', 'E102', 'Project Manager', '2024-02-01', '2024-10-01'),

('R103', 'P103', 'E103', 'Field Supervisor', '2024-03-01', '2024-09-01'),

('R104', 'P104', 'E104', 'Architect', '2024-04-10', '2024-12-10'),

('R105', 'P105', 'E105', 'Electrician', '2024-05-01', '2024-11-01'),

('R106', 'P106', 'E106', 'Site Engineer', '2024-06-01', '2024-12-15'),

('R107', 'P107', 'E107', 'Carpenter', '2024-07-01', '2024-12-31'),

('R108', 'P108', 'E108', 'Plumber', '2024-08-01', '2025-02-01'),

('R109', 'P109', 'E109', 'Mechanical Engineer', '2024-09-01', '2025-03-01'),

('R110', 'P110', 'E110', 'Civil Engineer', '2024-10-01', '2025-04-01'),

('R111', 'P111', 'E111', 'Construction Manager', '2024-11-01', '2025-05-01'),

('R112', 'P112', 'E112', 'Surveyor', '2024-12-01', '2025-06-01'),

('R113', 'P113', 'E113', 'Architect', '2025-01-01', '2025-07-01'),

('R114', 'P114', 'E114', 'Structural Engineer', '2025-02-01', '2025-08-01'),

('R115', 'P115', 'E115', 'Safety Officer', '2025-03-01', '2025-09-01'),

('R116', 'P116', 'E116', 'Interior Designer', '2025-04-01', '2025-10-01'),

('R117', 'P117', 'E117', 'Estimator', '2025-05-01', '2025-11-01'),

('R118', 'P118', 'E118', 'Project Coordinator', '2025-06-01', '2025-12-01'),

('R119', 'P119', 'E119', 'Electrical Engineer', '2025-07-01', '2026-01-01'),

('R120', 'P120', 'E120', 'HVAC Technician', '2025-08-01', '2026-02-01'),

('R121', 'P121', 'E121', 'Drafter', '2025-09-01', '2026-03-01'),

('R122', 'P122', 'E122', 'Foreman', '2025-10-01', '2026-04-01'),

('R123', 'P123', 'E123', 'Laborer', '2025-11-01', '2026-05-01'),

('R124', 'P124', 'E124', 'Welder', '2025-12-01', '2026-06-01'),

('R125', 'P125', 'E125', 'Environmental Engineer', '2026-01-01', '2026-07-01'),

('R126', 'P101', 'E101', 'Structural Engineer', '2026-02-01', '2026-08-01'),

('R127', 'P102', 'E102', 'Project Manager', '2026-03-01', '2026-09-01'),

('R128', 'P103', 'E103', 'Field Supervisor', '2026-04-01', '2026-10-01'),

('R129', 'P104', 'E104', 'Architect', '2026-05-01', '2026-11-01'),

('R130', 'P105', 'E105', 'Electrician', '2026-06-01', '2026-12-01')

Complex Queries

1.) **Clients with the Most Overdue Payments**

```
SELECT Clients.ClientID, Clients.Name AS ClientName,
    COUNT(Invoices.InvoiceID) AS OverdueInvoices,
    SUM(Invoices.Amount) AS TotalOverdue FROM Clients
JOIN Projects ON Clients.ClientID = Projects.ClientID
JOIN Invoices ON Projects.ProjectID = Invoices.ProjectID
WHERE Invoices.Status = 'Overdue'
GROUP BY Clients.ClientID, Clients.Name

ORDER BY TotalOverdue DESC;
```

```
Query 1
Limit to 1000 rows
348  -- 1)Clients with the Most Overdue Payments
349  SELECT Clients.ClientID, Clients.Name AS ClientName,COUNT(Invoices.InvoiceID) AS OverdueInvoices,
350  SUM(Invoices.Amount) AS TotalOverdue FROM Clients
351  JOIN Projects ON Clients.ClientID = Projects.ClientID
352  JOIN Invoices ON Projects.ProjectID = Invoices.ProjectID
353  WHERE Invoices.Status = 'Overdue'
354  GROUP BY Clients.ClientID, Clients.Name
355  ORDER BY TotalOverdue DESC;
```

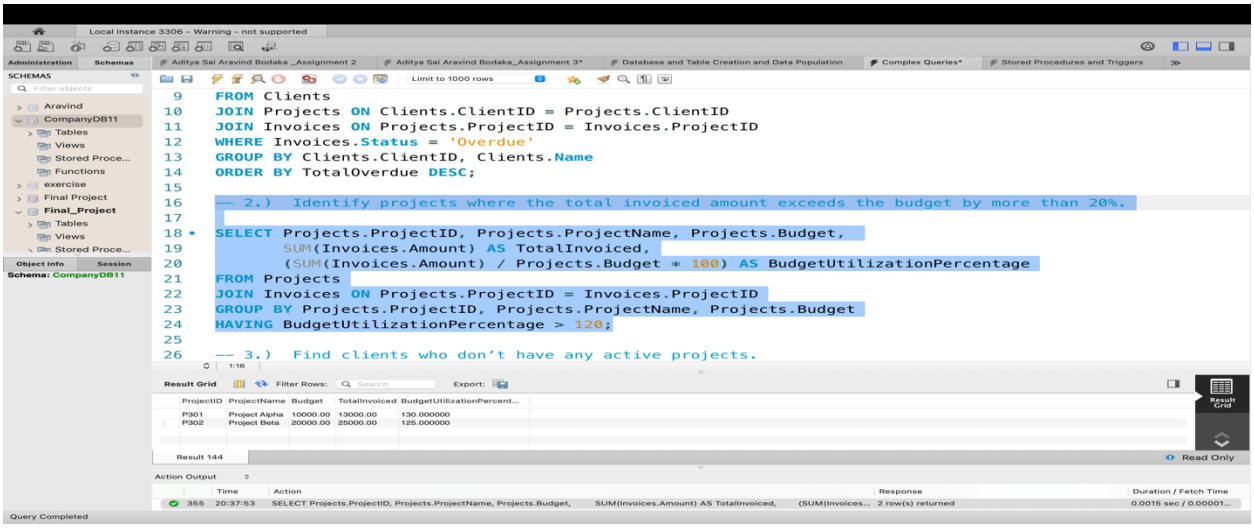
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ClientID	ClientName	OverdueInvoices	TotalOverdue
C103	Pinnacle Properties	1	85000.00
C109	Riverfront Holdings	1	45000.00
C115	Highland Constructors	1	32000.00
C102	EcoHomes LLC	1	24000.00
C123	Maplewood Homes	1	14000.00

This query helps to find the past-due invoices, this query determines which clients have the largest outstanding payments. By determining the overall number and amount owing, it helps select important clients for follow-up, such as "EcoHomes LLC" ($85,000), enhancing cash flow and financial stability.

2.) Identify projects where the total invoiced amount exceeds the budget by more than 20%.

```
SELECT Projects.ProjectID, Projects.ProjectName, Projects.Budget,
    SUM(Invoices.Amount) AS TotalInvoiced,
    (SUM(Invoices.Amount) / Projects.Budget * 100) AS BudgetUtilizationPercentage
FROM Projects
JOIN Invoices ON Projects.ProjectID = Invoices.ProjectID
GROUP BY Projects.ProjectID, Projects.ProjectName, Projects.Budget
HAVING BudgetUtilizationPercentage > 120;
```

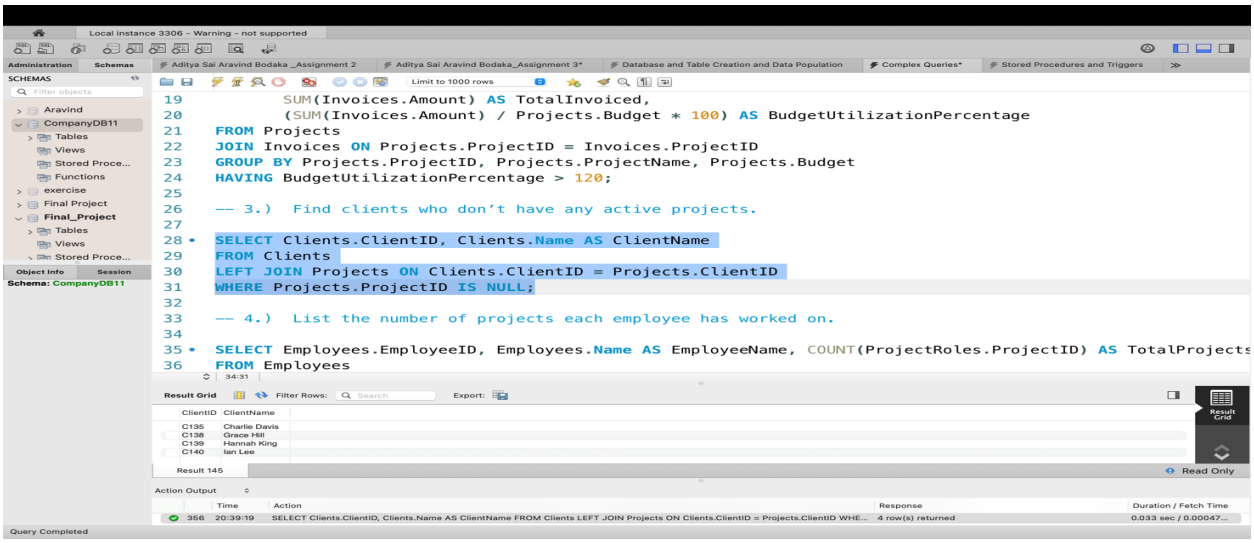


This query links the Projects and Invoices tables to determine budget usage and finds projects where total invoiced amounts are more than 20% over budget. Projects are

identified when their invoices exceed 120% of their budget. By keeping an eye on budget overruns, this promotes improved financial management and prompt remedial action to prevent cost increases.

3.) Find clients who don't have any active projects.

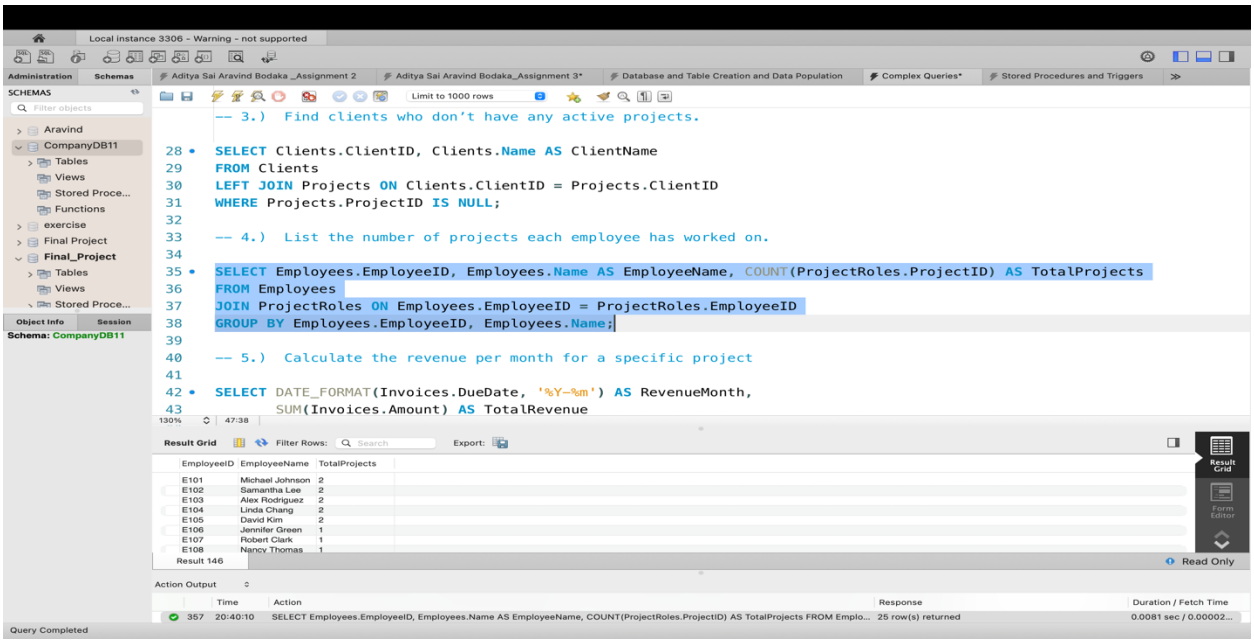
```
SELECT Clients.ClientID, Clients.Name AS ClientName
FROM Clients
LEFT JOIN Projects ON Clients.ClientID = Projects.ClientID
WHERE Projects.ProjectID IS NULL;
```



This query identifies clients with no active projects by performing a LEFT JOIN between the Clients and Projects tables and filtering for NULL in Projects.ProjectID. It helps the company target inactive clients for re-engagement or marketing opportunities, supporting client retention and business growth.

4.) List the number of projects each employee has worked on.

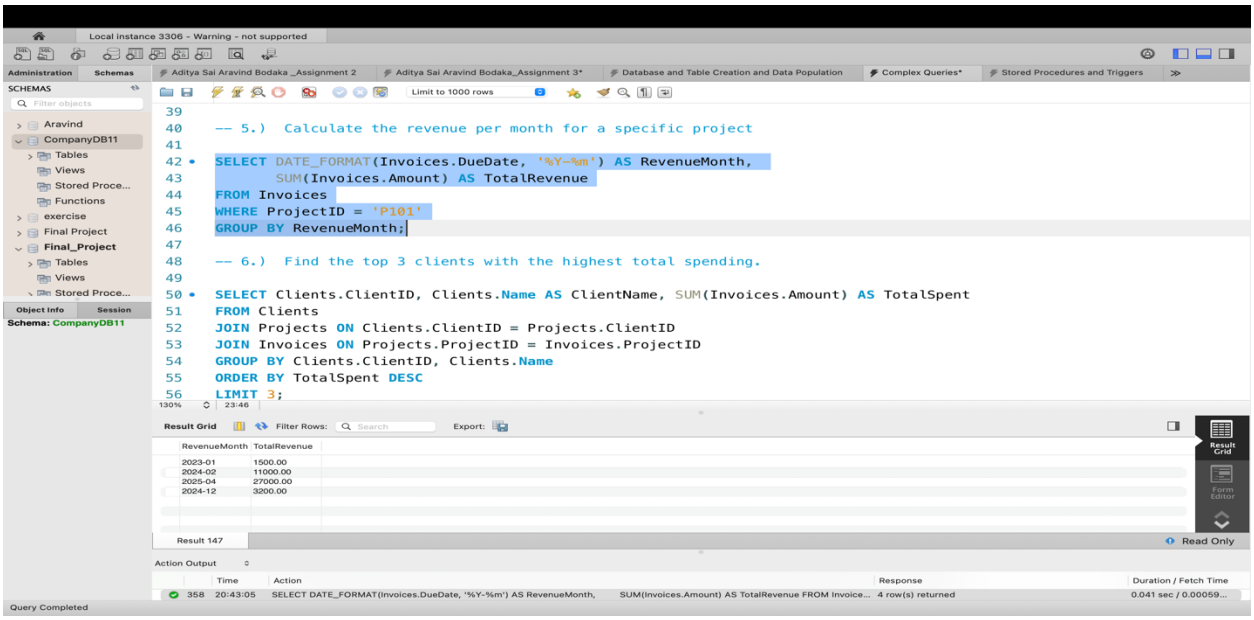
```
SELECT Employees.EmployeeID, Employees.Name AS EmployeeName,
COUNT(ProjectRoles.ProjectID) AS TotalProjects
FROM Employees
JOIN ProjectRoles ON Employees.EmployeeID = ProjectRoles.EmployeeID
GROUP BY Employees.EmployeeID, Employees.Name;
```



This query joins the Employees and ProjectRoles tables and groups by employee to determine how many projects each employee has worked on. It facilitates the tracking of employee participation in projects, guarantees effective resource distribution, and identifies high-achieving team members for improved workforce management.

5.) Calculate the revenue per month for a specific project

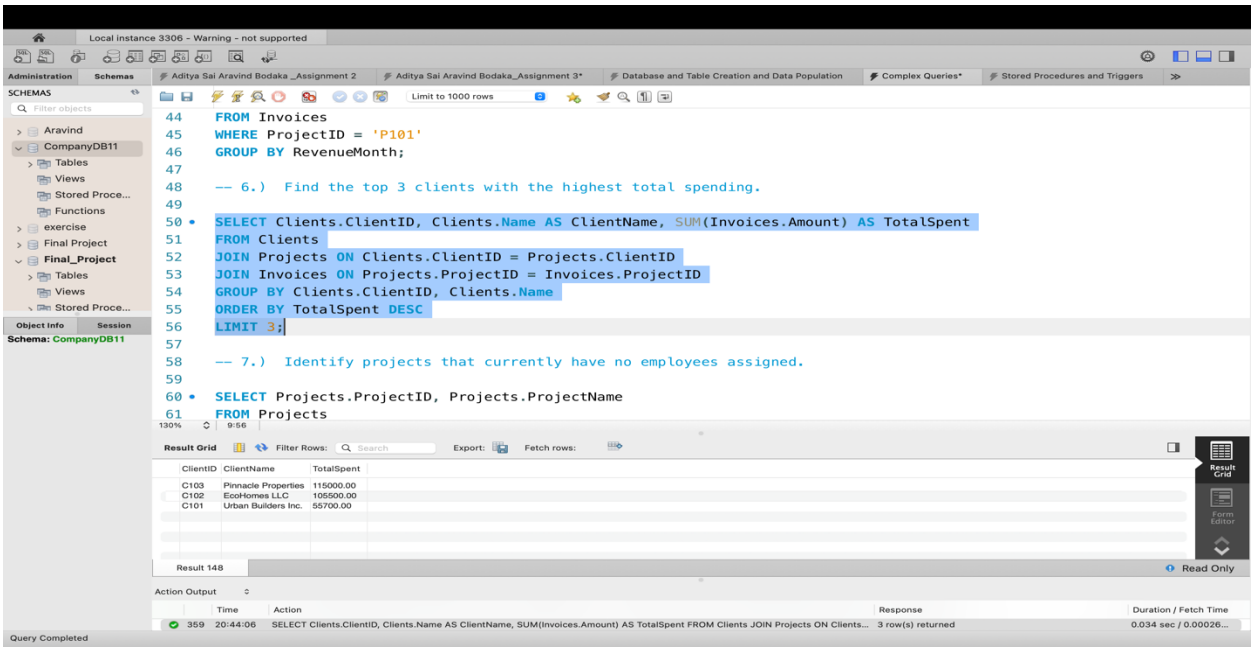
```
SELECT DATE_FORMAT(Invoices.DueDate, '%Y-%m') AS RevenueMonth,
    SUM(Invoices.Amount) AS TotalRevenue
FROM Invoices
WHERE ProjectID = 'P101'
GROUP BY RevenueMonth;
```



This query adds up invoice amounts (Invoices.Amount) by month (RevenueMonth) to determine monthly revenue for a particular project. It supports budget planning and forecasting by tracking a project's financial performance over time and offering insights into cash flow trends.

6.) Find the top 3 clients with the highest total spending.

```
SELECT Clients.ClientID, Clients.Name AS ClientName, SUM(Invoices.Amount) AS TotalSpent
FROM Clients
JOIN Projects ON Clients.ClientID = Projects.ClientID
JOIN Invoices ON Projects.ProjectID = Invoices.ProjectID
GROUP BY Clients.ClientID, Clients.Name
ORDER BY TotalSpent DESC
LIMIT 3;
```

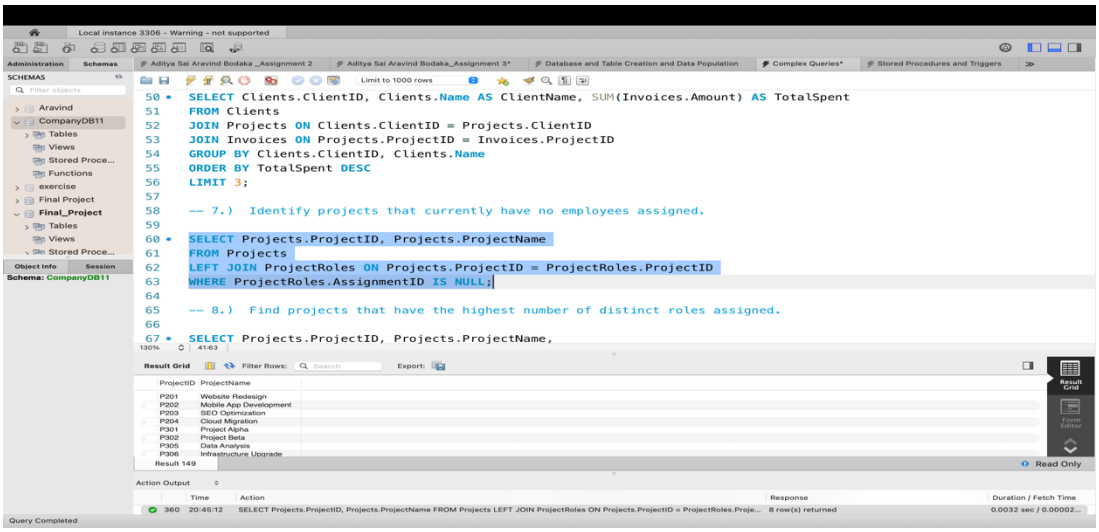


This query adds together the invoice amounts associated with the projects of the top three clients to determine who has spent the most overall. By helping the business identify its most valued customers, it may improve relationship management and strategic focus to optimize income and future prospects.

7.) Identify projects that currently have no employees assigned.

SELECT Projects.ProjectID, Projects.ProjectName FROM Projects

LEFT JOIN ProjectRoles ON Projects.ProjectID = ProjectRoles.ProjectID
WHERE ProjectRoles.AssignmentID IS NULL;

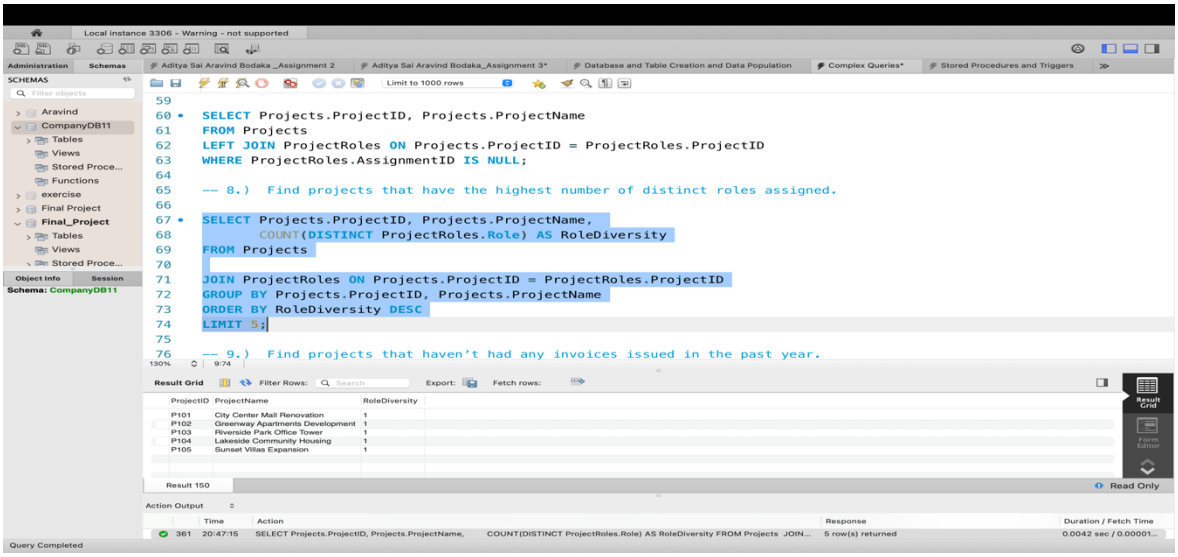


This query filters for NULL in AssignmentID and uses a LEFT JOIN between Projects and ProjectRoles to find projects without any employees assigned. It enhances resource allocation and

project management effectiveness by ensuring that all projects have a sufficient number of employees.

8.) **Find projects that have the highest number of distinct roles assigned.**

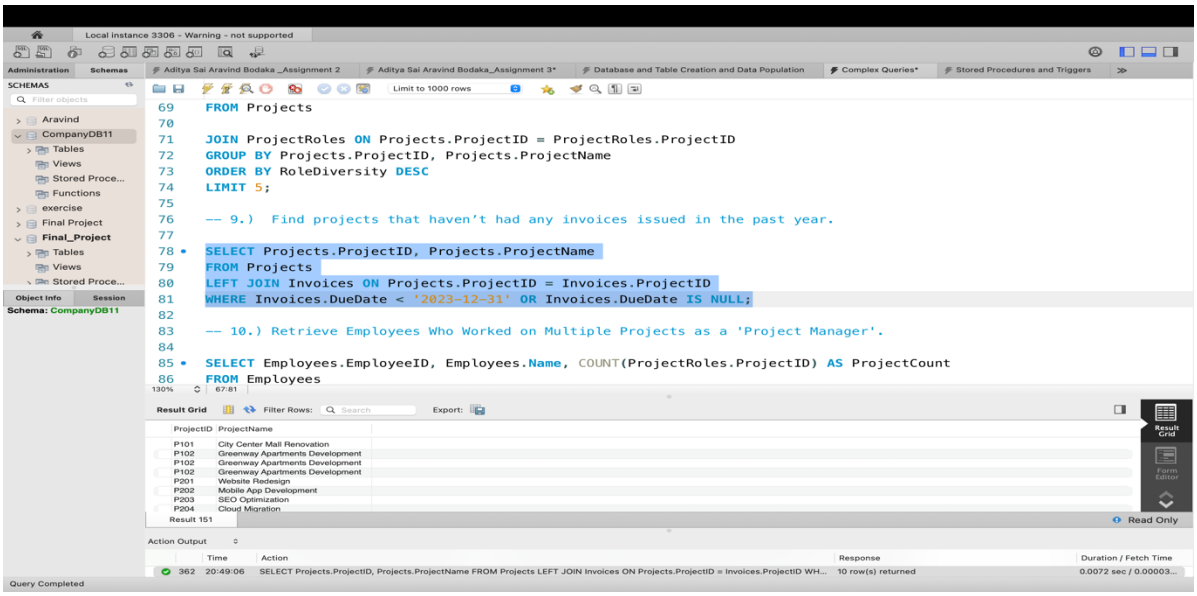
```
SELECT Projects.ProjectID, Projects.ProjectName,
    COUNT(DISTINCT ProjectRoles.Role) AS RoleDiversity
FROM Projects
JOIN ProjectRoles ON Projects.ProjectID = ProjectRoles.ProjectID
GROUP BY Projects.ProjectID, Projects.ProjectName
ORDER BY RoleDiversity DESC
LIMIT 5;
```



This query counts the different roles assigned to each project in the ProjectRoles table to see which projects have the most diversified roles assigned. By identifying complex initiatives that call for specialist knowledge and maintaining balanced talent distribution, it aids in evaluating role variation across projects.

9.) **Find projects that haven't had any invoices issued in the past year.**

```
SELECT Projects.ProjectID, Projects.ProjectName
FROM Projects
LEFT JOIN Invoices ON Projects.ProjectID = Invoices.ProjectID
WHERE Invoices.DueDate < '2023-12-31' OR Invoices.DueDate IS NULL;
```



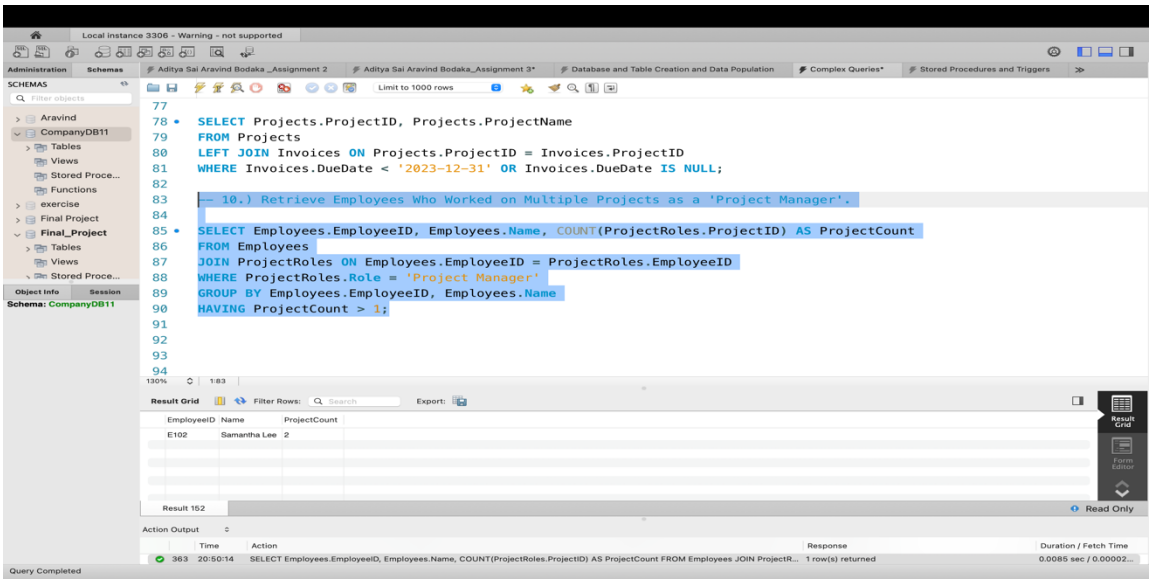
This query finds for NULL in Invoices to find projects that haven't had any invoices made in the previous year. Date of due or dates before to '2023-12-31'. By tracking stopped or inactive projects, it helps the business better monitor finances and deal with delays.

10.) Retrieve Employees Who Worked on Multiple Projects as a 'Project Manager'.

SELECT Employees.EmployeeID, Employees.Name, COUNT(ProjectRoles.ProjectID) AS ProjectCount FROM Employees JOIN ProjectRoles ON Employees.EmployeeID = ProjectRoles.EmployeeID WHERE ProjectRoles.Role = 'Project Manager'

GROUP BY Employees.EmployeeID, Employees.Name

HAVING ProjectCount > 1;



This query counts the projects that employees have been given in the ProjectRoles table to retrieve those who have worked on many projects as a "Project Manager." Finding seasoned project managers aids in resource planning and identifies important team members.

Stored Procedures and Triggers file(s):

Stored Procedures

1.) This procedure calculates and updates the total invoiced amount for a specific project.

```
DELIMITER //
create procedure UpdateProjectInvoicedAmount(in projectID varchar(10))
begin
    update Projects
    set Budget = (select sum(Amount) from Invoices where ProjectID = projectID)
    where ProjectID = projectID;
end //
DELIMITER ;

/*SELECT ProjectID, Budget FROM Projects WHERE ProjectID = 'P101';
SELECT ProjectID, SUM(Amount) AS TotalInvoiced FROM Invoices WHERE ProjectID = 'P101';

UPDATE Invoices
SET Amount = Amount+500
WHERE InvoiceID = 'INV101' AND ProjectID = 'P101';

CALL UpdateProjectInvoicedAmount('P101');
*/
```

2.) This procedure marks invoices as overdue if their due date has passed and they are not yet paid.

```
DELIMITER //
create procedure MarkOverdueInvoices()
begin
    update Invoices
    set Status = 'Overdue'
    where DueDate < curdate() and Status != 'Paid';
end //
DELIMITER ;
/*
INSERT INTO Invoices (InvoiceID, ProjectID, Amount, DueDate, Status)
VALUES ('INV999', 'P101', 1500, '2025-01-01', 'Pending');

UPDATE Invoices
SET DueDate = '2023-01-01'
WHERE InvoiceID = 'INV999';

CALL MarkOverdueInvoices();

SELECT InvoiceID, DueDate, Status FROM Invoices WHERE InvoiceID = 'INV999';
select * from invoices;
*/
```

TRIGGERS

1.) This trigger automatically marks an invoice as overdue when the due date is updated to a past date.

```
DELIMITER //
```

```
CREATE TRIGGER SetOverdueStatus
BEFORE UPDATE ON Invoices
FOR EACH ROW
BEGIN
    IF NEW.DueDate < CURDATE() AND NEW.Status != 'Paid' THEN
        SET NEW.Status = 'Overdue';
    END IF;
END //
DELIMITER ;
```

2.) Automatically Set Default Invoice Status.

```
DELIMITER //
CREATE TRIGGER SetDefaultInvoiceStatus
BEFORE INSERT ON Invoices
FOR EACH ROW
BEGIN
    IF NEW.Status IS NULL OR NEW.Status = '' THEN
        SET NEW.Status = 'Pending';
    END IF;
END //
DELIMITER ;
```

Database Optimization

Indexing

The efficiency of database queries can be greatly enhanced by indexing, particularly when handling intricate processes that involve numerous tables, filters, and aggregations. In the provided

query, which finds clients with past-due invoices and determines their total amount owed, indexing is essential for assuring effective data retrieval and cutting down on query execution time.

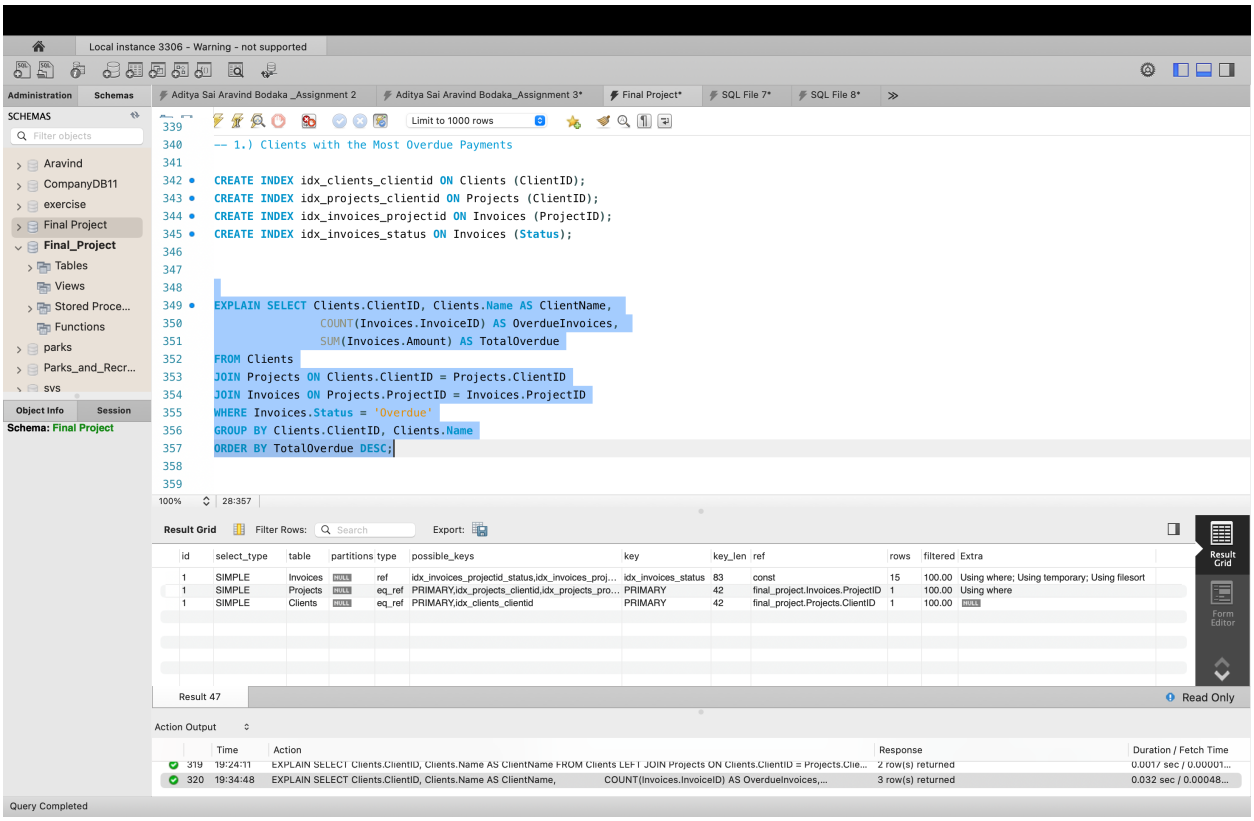
The database must run a full table scan without indexes, going over each row one after the other to locate records that match. As the database size increases, this method becomes less and less effective. The database may retrieve rows in logarithmic time instead of linear time thanks to indexes, which create a sorted data structure that maps column values to their corresponding row locations.

The database pulls information from the Clients, Projects, and Invoices tables in the provided query. ClientID between Clients and Projects and ProjectID between Projects and Invoices are the foreign key relationships that are used to join these tables. A filter condition is also applied by the query to the Invoices table's Status column, choosing records with the status "Overdue." In order to determine the number and total of past-due invoices for every client, it lastly carries out grouping and aggregation. Next, it sorts the invoices according to the total amount that is past due.

Indexing significantly enhances the performance of this query. First, single-column indexes on the foreign key columns—ClientID in Clients and Projects, and ProjectID in Projects and Invoices—optimize the join operations. These indexes allow the database to efficiently locate matching rows between the tables, avoiding the need to compare every row from one table with every row from the other. Similarly, an index on the Status column in the Invoices table accelerates the filtering operation. Instead of scanning all rows to find those with a status of "Overdue," the database can quickly locate the relevant rows using the index.

1.) Clients with the Most Overdue Payments

Indexes are like shortcuts in a database that make searching for data faster and more efficient. Before using indexes, the query had to scan all rows in the `Clients`, `Projects`, and `Invoices` tables, which took a lot of time and effort, especially with large amounts of data. This did tasks like filtering overdue invoices, joining tables, and sorting results very slowly.



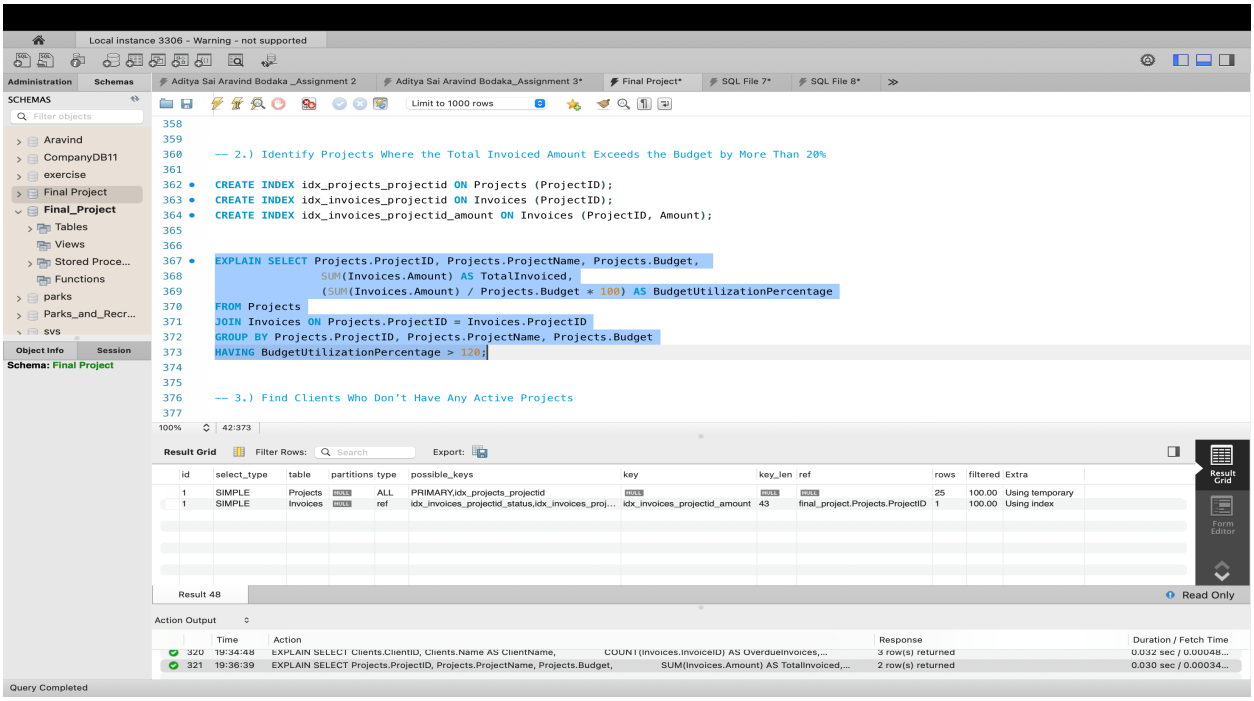
To fix this, indexes were added to the columns most used in the query, like `ClientID`, `ProjectID`, and `Status`. These indexes helped the database quickly find only the rows it needed.

The query also grouped and sorted results, like finding the total overdue amount for each client. With indexes, this was faster because the database could access organized data directly. After adding indexes, the query ran much faster, scanned fewer rows, and avoided unnecessary work like using temporary tables. This shows how indexing can turn a slow query into a quick and efficient one, saving time and resources.

2.) Identify Projects Where the Total Invoiced Amount Exceeds the Budget by More Than 20%

The database performed **full table scans** on the Invoices and Projects tables without the indexing. This meant that every row in both tables had to be checked to calculate the total invoiced amount and determine which projects exceeded their budgets by more than 20%. This process was

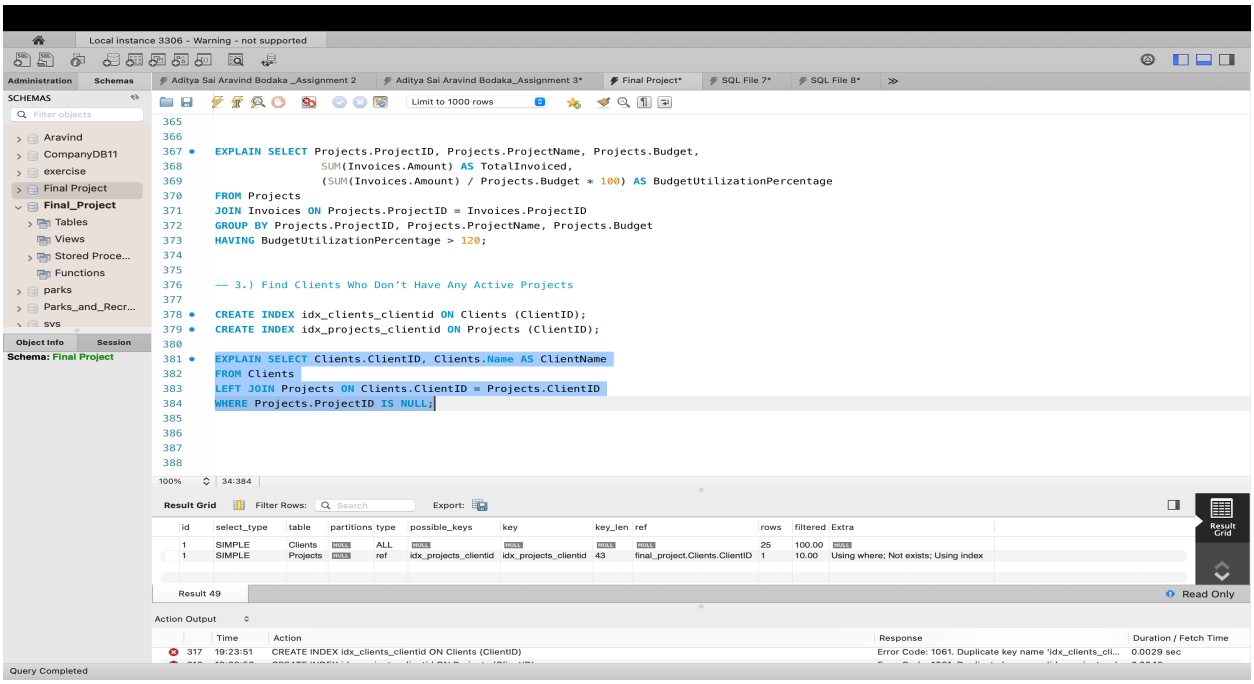
inefficient and slow, especially as the size of the tables grew larger, because the database lacked any shortcuts to quickly retrieve relevant data.



But after the introduction of indexes, particularly a **composite index** on the Invoices table for columns ProjectID and Amount, the query became significantly faster. The indexed ProjectID enabled quicker joins between the Invoices and Projects tables, while the indexed Amount column sped up the aggregation for calculating the total invoiced amount. Additionally, indexing on Projects. ProjectID helped optimize grouping operations, making the entire process of filtering projects that exceeded their budgets more efficient.

3.) Find Clients Who Don't Have Any Active Projects

The table without the indexes, the database performed full table scans on both the Clients and Projects tables to identify clients without any active projects. This meant that every row in the Clients table had to be compared with every row in the Projects table to find matching records.



With the introduction of indexes, particularly idx_clients_clientid on the Clients.ClientID column and idx_projects_clientid on the Projects.ClientID column, the query performance improved significantly. The LEFT JOIN operation became more efficient as the database could quickly locate matching rows between the Clients and Projects tables. Additionally, the filtering condition (WHERE Projects.ProjectID IS NULL) was processed faster because the indexed columns helped reduce the number of rows scanned. These optimizations resulted in faster query execution and lower resource usage.

Data Security and Backup

Data Security Plan

The database implements **Role-Based Access Control (RBAC)** and other measures to ensure data integrity and prevent unauthorized access.

User Roles and Permissions

To protect the integrity of the construction project database and ensure only authorized access, implement Role-Based Access Control (RBAC). Different roles are designed to align with database entities such as Clients, Projects, Invoices, Suppliers, and Employees. These roles ensure users access only the data they need for their job functions.

Roles and Permissions:

Administration: Has full control over all tables, including modifying records in Clients, Projects, Invoices, and Employees. They can add, update, or delete data and manage user permissions.

Sales Manager: Can view and manage data in Clients, Projects, and Invoices for tracking client activities and payments. They can retrieve overdue invoices or monitor project-client connections.

Inventory Manager: Able to control material stock and supplier ratings by accessing Suppliers and Inventory. Once supplies are received, they can adjust inventory quantities.

Project Manager: Able to assign staff to tasks, monitor the status of ongoing projects, and view and manage projects and project roles.

Employees: Not allowed to change any data; just able to view their individual assignments in ProjectRoles.

Mechanisms of Security: To stop unwanted access, use authentication techniques like session timeouts and two-factor authentication. Implement stringent password regulations and regular audits to protect system integrity.

Data Backup Plan

A strong backup plan is essential to guarantee data recovery and reduce interruptions brought on by unanticipated circumstances like hardware malfunctions, cyberattacks, or inadvertent deletions. The database contains important entities that are necessary for company operations, such as contracts, invoices, and projects.

Backup Strategy:

Full Backups: Weekly backups of the complete database, comprising ProjectRoles, Clients, Invoices, Inventory, and Projects. Guarantees that all data can be recovered in the event of a significant system failure.

Incremental Backups: Daily backups of tables like Projects, Invoices, and ProjectRoles that are updated often. It saves only the changes made since the last backup, reducing storage requirements.

Automated Backups: To ensure consistency without requiring human interaction, schedule backups using database tools or scripts.

Restore Strategy

In the event of data loss, a well-organized restoration strategy guarantees that the database can be promptly restored.

Validation and Testing: To make sure backups are working and data integrity is maintained, verify the restore procedure on a regular basis. To get ready for future calamities, practice recovery scenarios.

Prioritization: To reduce interruption, important tables like Projects, Invoices, and Clients should be rebuilt first. Depending on operational requirements, other tables, such as Suppliers or Employees, may be restored later.

Data Preservation

Make sure historical data is accessible for audits or compliance needs by keeping backups for a minimum of six months.

To fulfill contractual or regulatory requirements, archive monthly backups for long-term storage.

To cut storage expenses and stop illegal access to out-of-date data, safely remove old backups.